Convergence et taux de convergence d'un algorithme fourmi simple

Amine Boumaza, Bruno Scherrer

Convergence et taux de convergence d'un algorithme fourmi simple

Amine Boumaza — Bruno Scherrer

Equipe MAIA, LORIA
Campus Scientifique BP 239
54506 Vandœuvre-lès-Nancy CEDEX, France
{amine.boumaza, bruno.scherrer}@loria.fr

RÉSUMÉ. *Nous présentons dans cet article un algorithme fourmi simple pour résoudre le problème du fourragement dans le cas discret. Nous illustrons le modèle proposé à l'aide de simulations et nous faisons une analyse complète de convergence: nous démontrons que la population de fourmis calcule la solution de problèmes de contrôle optimal et qu'elle converge dans un certain sens. Nous étudions le taux de convergence du modèle en fonction de la taille de la population et donnons des arguments expérimentaux et théoriques qui suggèrent que ce taux de convergence est super-linéaire en fonction du nombre d'agents. En outre, nous expliquons comment ce modèle peut être étendu au cas où l'espace est continu et pour résoudre des problèmes de contrôle optimal en général. Nous expliquons qu'une telle approche peut être appliquée à tout problème qui implique le calcul du point fixe d'une contraction. Ceci permet de concevoir une grande classe d'algorithmes de type fourmis bien compris formellement.*

ABSTRACT. *We present a simple ant model that solves a discrete foraging problem. We describe simulations and provide a complete convergence analysis: we show that the ant population computes the solution of some optimal control problems and converges in some well defined sense. We discuss the rate of convergence with respect to the number of ants: we give experimental and theoretical arguments that suggest that this convergence rate is superlinear with respect to the number of agents. Furthermore, we explain how this model can be extended to the case where the state space is continuous, and in order to solve optimal control problems in general. We argue that such an approach can be applied to any problem that involves the computation of the fixed point of a contraction mapping. This allows to design a large class of formally well understood ant-like algorithms.*

MOTS-CLÉS : *Système multi-agents, algorithmes fourmis, planification multi-agents*

KEYWORDS: *Multiagent Systems, Ant Algorithms, Multiagent planning*

1re soumission à *RIA–Modèles multi-agents pour des environnements complexes*, le 17 novembre 2006

1. Introduction

Le terme *swarm intelligence* (Beni *et al.*, 1989; Bonabeau *et al.*, 1999), en français « l'intelligence en essaim », a été introduit dans les années 90 comme une nouvelle approche, souvent bio-inspirée, pour la résolution de problèmes. La source d'inspiration de cette approche est le comportement des insectes sociaux : une population d'agents simples interagissant et communiquant indirectement à travers leur environnement constitue un algorithme *massivement parallèle* pour résoudre une tâche donnée (par exemple le fourragement, l'attroupement ou *flocking*, la division de tâches, la capture de proies etc...). À propos du domaine des réseaux de neurones, une autre approche bio-inspirée pour la résolution *massivement parallèle* de problèmes, Kohonen a écrit en 1988 (Kohonen, 1988)[1] :

> « *Une des tâches fondamentales de cette nouvelle science des réseaux de neurones est de démontrer par des analyses mathématiques, des simulations sur ordinateur, et même à travers des systèmes sensoriels ou de commande artificiels qu'il est possible de mettre en application des fonctions de traitement de l'information massivement parallèles en utilisant des composants dont les principes ne sont pas mystérieux mais déjà familier de l'informatique, des sciences de la communication, et de l'automatique. Il n'y a rien dans le domaine des réseaux de neurones qui ne soit déjà connu ou suggéré, au moins en principe, par d'autres théories existantes.* »

La motivation de cet article est d'exprimer quelquechose de similaire au sujet de l'intelligence en essaim. Nous allons présenter un algorithme fourmi et montrer que des résultats des domaines "parallélisme" et "processus stochastiques" permettent de le comprendre et de l'analyser en profondeur.

L'approche classique pour la résolution de problèmes et l'intelligence en essaim peuvent être vues comme deux manières alternatives de faire de la résolution de problèmes. En quoi ces approches sont-elles si différentes ? Comme évoqué par Sutton (Sutton, 1988) (qui parle de la relation entre l'apprentissage artificiel « moderne » et le contrôle intelligent « classique »[2]), on pourrait

> « *caractériser la différence comme étant lié au dilemme familier entre obtenir un résultat clair, rigoureux d'un côté, et explorer les systèmes les plus intéressants, puissants de l'autre.* »

---

1. Dans le texte original : « One of the fundamental tasks of this new "neural networks" science is to demonstrate by mathematical analyses, computer simulations, and even working artificial sensory and control systems that it is possible to implement massively parallel information-processing functions using components the principles of which are not mysterious but already familiar from computer technology, communication science, and control engineering. There is nothing in the "neural network" area which were not known, in principle at least, from constructs already in use or earlier suggested. »

2. Dans le texte original : « [We could] characterize the split as having to do with the familiar dilemma of choo sing between obtaining clear, rigorous results on the one hand, and exploring the most interesting, powerful systems on the other. »

De manière très schématique, la recherche sur l'intelligence en essaim est souvent basée sur une méthodologie expérimentale pour étudier des problèmes difficiles alors que la résolution de problèmes par l'ingénierie classique se base sur des preuves théoriques de convergence pour des problèmes « jouets ». La question qui nous intéresse ici n'est pas de juger ces approches (elles sont clairement complémentaires) mais d'essayer de tisser des liens entre les deux.

Dans la littérature, il existe peu de travaux qui tentent de faire ce lien. Une exception intéressante concerne l'optimisation par colonies de fourmis (ACO) pour les problèmes combinatoires, probablement la méta-heuristique la plus connue en intelligence par essaim. Les travaux théoriques sur l'ACO on été récemment analysés dans (Dorigo *et al.*, 2005). Les auteurs y présentent des théorèmes pour la « convergence en valeurs » (l'ACO est garantie de trouver une solution optimale en un temps fini avec une probabilité qui peut être arbitrairement proche de 1), et pour la « convergence en solution » (avec un paramètre décroissant d'exploration bien conçu, l'ACO converge vers une solution optimale presque sûrement). Les auteurs font aussi le lien entre l'ACO et d'autre approches « plus standards » comme celles de descente de gradient stochastique ou d'entropie croisée. Nous renvoyons le lecteur intéressé à (Dorigo *et al.*, 2005) pour plus de détails.

Dans cet article, nous revenons à l'inspiration originale des algorithmes fourmis, où une population d'agents simples (qui peut être vue comme imitant le comportement de fourmis réelles) résout efficacement le problème du fourragement. Nous allons décrire quelques modèles massivement parallèles à base d'agents dont on peut prouver qu'ils résolvent le problème du fourragement dans un sens bien défini. De plus, nous donnerons une analyse théorique du taux de convergence en fonction du nombre d'agents.

Le reste de l'article est organisé comme suit. Dans une première partie (section 2) nous décrivons un modèle discret de fourmis et présentons quelques simulations qui montrent que la dynamique de la population converge dans un certain sens. Nous présentons des simulations qui suggèrent qu'il y a un taux de convergence super-linéaire. La seconde partie (section 3) donne une preuve de la convergence et donne des éclaircissements sur l'influence des paramètres du modèle présenté. Dans une troisième partie (section 4) nous analysons le taux de convergence et donnons des arguments qui expliquent pourquoi on observe un taux de convergence super-linéaire. La section 5 montre comment on peut étendre le modèle discret au cas du fourragement dans un espace continu et explique dans quelle mesure on peut adapter l'analyse (convergence et taux de convergence) au cas continu. Finalement, nous discutons (section 6) de la portée de notre modèle et des relations qu'il entretient avec d'autres modèles de la littérature.

2. Un modèle simple fourmis

Dans cette partie, nous décrivons un modèle simple de fourmis qui permet de résoudre le problème du fourragement. Nous exhibons des simulations qui montrent que les fourmis finissent par transporter la nourriture depuis une source vers leur nid, et nous présentons des données expérimentales qui montrent que l'efficacité de ce modèle distribué est super-linéaire en le nombre de fourmis.

2.1. *Description du modèle*

Considérons un ensemble de m fourmis artificielles qui évoluent sur les cellules d'une grille 2D[3] (l'environnement) et sur lesquelles elles mettent à jour des "taux de phéromones". Chaque cellule s de la grille stocke deux valeurs réelles, correspondant à deux types de phéromone : $V_1(s)$ et $V_2(s)$. Sur la grille, on distingue quatre types de cellules : une cellule correspond au *nid*, une à la *source de nourriture*, un certain nombre de cellules sont *indésirables* et le reste des cellules sont *libres*.

Une fourmi peut être dans deux états possibles : elle peut porter de la nourriture (état « chargée ») ou elle peut ne rien porter (état « pas chargée »). En évoluant dans l'environnement, l'état des fourmis change selon les règles naturelles suivantes : si une fourmi arrive à la cellule source de nourriture, son état devient « chargée », et si elle arrive au nid son état change à « pas chargée ». Quand une fourmi dans l'état « chargée » arrive au nid, un compteur d'« unités de nourriture » est incrémenté. Comme nous le verrons par la suite, ce compteur servira comme mesure de performance globale pour la population.

Nous décrivons maintenant la dynamique du modèle. Au départ,

– le compteur d'« unités de nourriture » est initialisé à zéro,

– les m fourmis sont initialisées arbitrairement (e.g. elles peuvent être toutes au nid, ou initialisées uniformément sur la grille),

– toutes les fourmis sont dans l'état « pas chargée »,

– les phéromones sont initialisées arbitrairement (e.g. 0, aléatoire, etc ...).

A chaque pas de temps, chaque fourmi fait deux choses :

– Elle met à jour localement les taux de phéromones $V_1(s)$ et $V_2(s)$ de sa cellule courante s en utilisant ceux des quatre cellules voisines (on écrit l'ensemble des voisins de s, $\mathcal{N}(s)$), c'est-à-dire en utilisant uniquement une information locale. La mise à jour des taux de phéromones requiert uniquement la connaissnace du maximum et de la moyenne des valeurs voisines : $\max_i(\mathcal{N}(s)) \triangleq \max_{s' \in \mathcal{N}(s)} V_i(s')$ et $\text{avg}_i(\mathcal{N}(s)) \triangleq \frac{1}{4} \sum_{s' \in \mathcal{N}(s)} V_i(s')$ où l'indice $i \in \{1, 2\}$ spécifie lequel des deux taux

3. Comme le lecteur comprendra dans la suite, des graphes plus complexes peuvent être utilisés, toutefois nous nous restreignions au cas d'une grille 2D pour simplifier la présentation.

de phéromones est considéré. Précisément, la mise à jour se fait selon les calculs suivants :

$$V_1(s) \leftarrow \begin{cases} -1 & \text{si } s \text{ est une cellule } \textit{indésirable} \\ 1 & \text{si } s \text{ est la } \textit{source de nourriture} \\ \beta\left(\alpha \max_1(\mathcal{N}(s)) + (1-\alpha)\text{avg}_1(\mathcal{N}(s))\right) & \text{sinon} \end{cases}$$

$$V_2(s) \leftarrow \begin{cases} -1 & \text{si } s \text{ est une cellule } \textit{indésirable} \\ 1 & \text{si } s \text{ is le } \textit{nid} \\ \beta\left(\alpha \max_2(\mathcal{N}(s)) + (1-\alpha)\text{avg}_2(\mathcal{N}(s))\right) & \text{sinon} \end{cases}$$

où $0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$ avec la condition que $\beta < 1$ quand $\alpha = 1$.

– Elle avance à une de ses cellules voisines :

- avec une probabilité ϵ ($0 \leq \epsilon \leq 1$), qu'on appellera le taux *exploration*, elle avance sur une cellule choisie aléatoirement uniformément sur ses quatre voisines

- avec une probabilité $1-\epsilon$, elle avance sur la cellule avec la plus grande valeur de phéromone de V_1 ou V_2 en fonction de son état : si elle n'est « pas chargée » alors elle utilise V_1 et si elle est « chargée » elle utilise V_2.

Le paramètre β peut être considéré comme un paramètre d'*évaporation* et prend typiquement des valeurs proches de 1. Le paramètre α, que nous appellerons paramètre de *bruit* pour des raisons qui seront explicitées plus tard, doit être fixé proche de 0 ou proche de 1. Deux cas particuliers de ce modèle correspondent aux choix de paramètres ($\alpha = 1$, $0 < \beta < 1$) et ($\alpha = 0$, $\beta = 1$). Dans le premier, l'équation de mise à jour des phéromones devient :

$$V_i(s) \leftarrow \beta \max_i(\mathcal{N}(s)) \qquad [1]$$

et dans le deuxième, l'équation devient :

$$V_i(s) \leftarrow \text{avg}_i(\mathcal{N}(s)) \qquad [2]$$

qui est une simple mise à jour linéaire. Comme il apparaîtra dans l'analyse, la question selon laquelle l'activité de l'ensemble des fourmis (mises à jour des pheromones et mouvements dans l'environnement) est faite de manière synchrone ou asynchrone n'a pas d'importance.

En résumé, pour identifier précisément une instance de ce modèle (et pour que le lecteur ait la possibilité de reproduire les expériences décrites plus loin), les données suivantes doivent être spécifiées :

– l'environnement : l'ensemble des cellules et leur type (nid, nourriture, indésirable, libre),

– le nombre m de fourmis,

– la manière d'initialiser les positions des fourmis et les valeurs des phéromones V_1 et V_2 sur l'environnement,

– le paramètre de bruit α, le paramètre d'évaporation β et le taux exploration ϵ.

Le modèle que nous venons de décrire est constitué d'un ensemble d'agents réactifs simples qui communiquent indirectement à travers l'environnement. En un sens, c'est un modèle plus simple que le modèle « classique » utilisé dans les meta-heuristiques de type ACO : dans notre cas les phéromones ne nécessitent pas d'être évaporées en *chaque* cellule de l'environnement (ce qui constitue un calcul lourd) ; dans notre modèle, une sorte d'évaporation est faite à travers le paramètre β lors de la mise à jour locale. Toutefois, nos fourmis utilisent *deux* taux de phéromones alors que les modèles « classiques » en utilisent généralement un ; comme cela apparaîtra clairement au lecteur, l'utilisation d'un deuxième taux de phéromone compense le fait que nos fourmis sont de agents complètement *réactifs* qui ne mémorisent pas le chemin de retour au nid (comme c'est le cas dans l'ACO).

2.2. *Simulations*

Dans cette partie nous illustrons le comportement du modèle que nous venons de présenter à l'aide de simulations. Nous allons observer que l'activité des fourmis converge dans un certain sens. Nous décrivons ensuite un protocole expérimental qui nous permet de mesurer le taux de convergence en fonction du nombre de fourmis. Nous commentons les résultats expérimentaux obtenus brièvement dans la mesure où une analyse approfondie est développée dans les sections 3 et 4.

Considérons une exécution typique de l'algorithme dans laquelle nous fixons les paramètres comme suit : $\epsilon = 0.8$, $\alpha = 0.7$, $\beta = 0.9999$ et la taille de la population est $m = 150$. La figure 1 montre des captures d'écran de l'exécution à différents moments. On y voit l'évolution des fourmis au cours du temps : initialisées au départ au nid, elles se déplacent dans l'environnement à la recherche de nourriture. Une fois que la source de nourriture est trouvée par une fourmi, plus de fourmis commence à se diriger vers la source et un chemin commence à se former entre le nid et la source. À mesure que le temps passe, une grande proportion de fourmis suit le chemin et la dynamique semble se stabiliser. En fonction de la valeur du paramètre α, on peut observer des chemins de formes différentes, voire dans certains cas aucun chemin. La figure 2 illustre la distribution asymptotique des fourmis pour des valeurs de α différentes : il existe des chemins sauf pour le cas où $\alpha = 0.6$.

Supposons qu'un chemin émerge entre le nid et la source de nourriture. Si l'on peut observer visuellement ce chemin, il est plus intéressant d'en faire une mesure objective. Ceci peut être fait via le compteur d'unités de nourriture introduit précédemment. En effet, on peut tracer la courbe de la quantité accumulée de nourriture rapportée au nid au cours du temps (voir la figure 3). Au bout d'un certain temps, on observe que la quantité accumulée de nourriture apportée au nid croît linéairement : ceci suggère que le comportement de fourragement des fourmis a convergé. Dans la section 3, nous fournirons des arguments théoriques qui caractérisent précisément cette convergence, et nous expliquerons en particulier pourquoi il n'y a pas de chemin pour certaines valeurs de α.

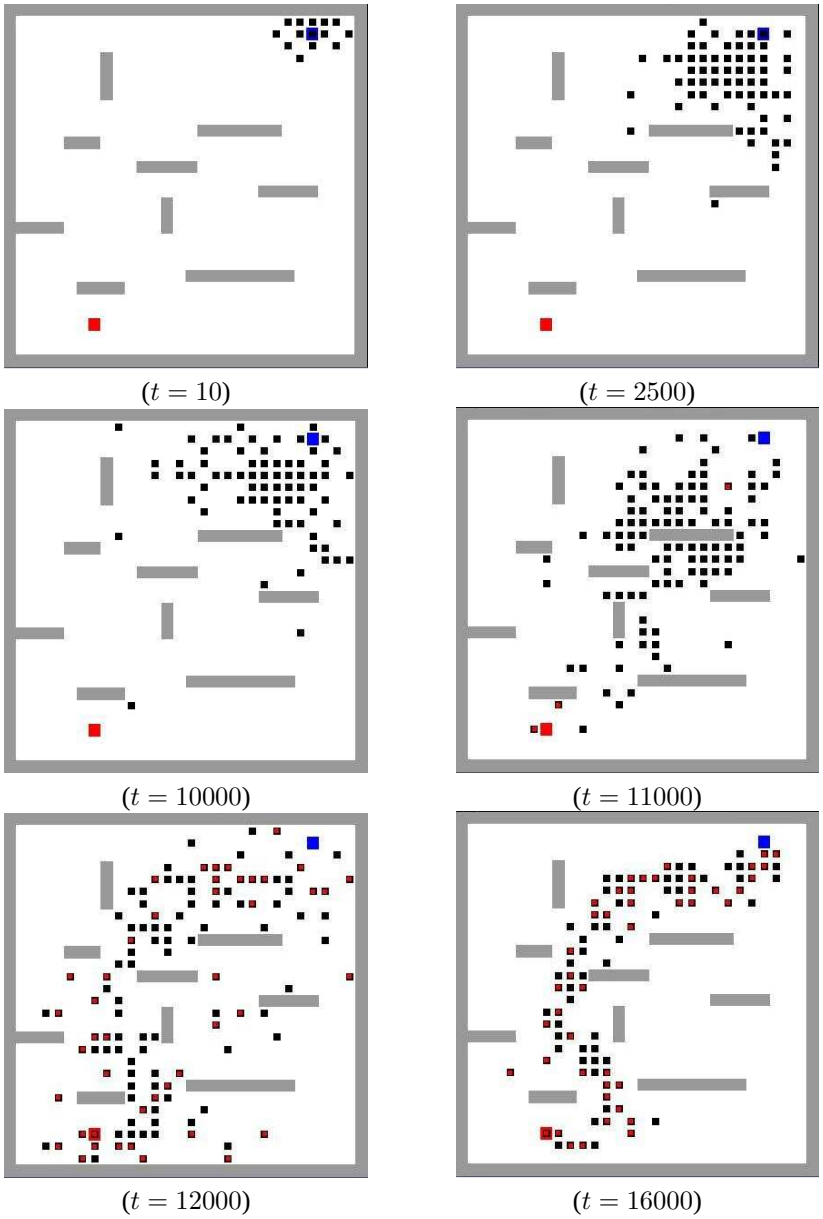

Figure 1. *Capture d'écran du modèle de fourmi : les petits carrés représentent des fourmis se déplaçant entre la source de nourriture (en bas à gauche) et le nid (en haut à droite).*

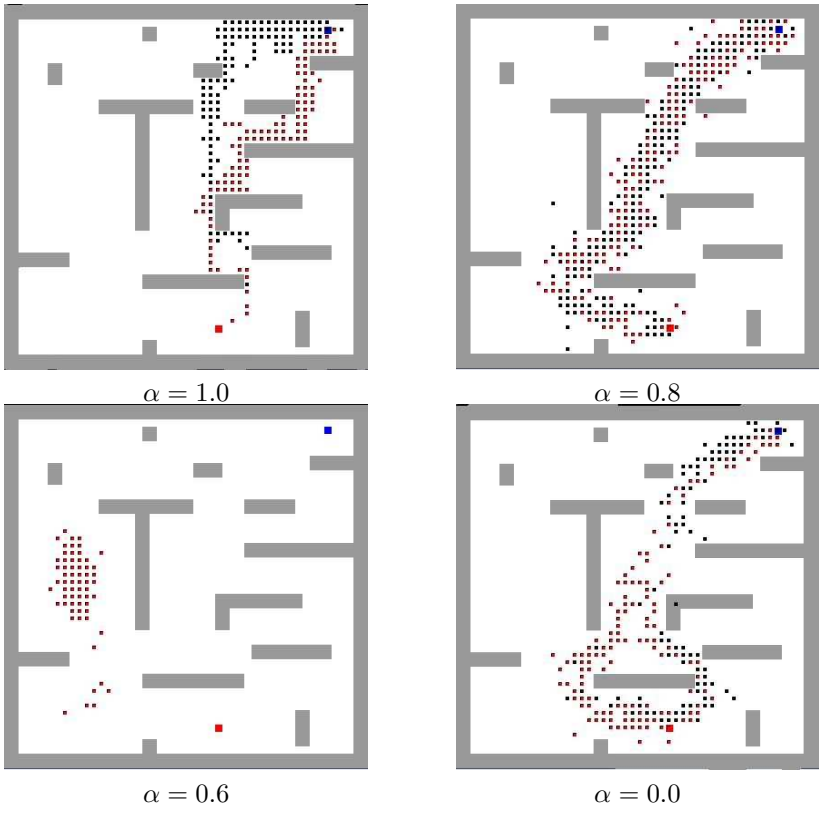

Figure 2. *Distribution limite des fourmis en fonction de α.*

La courbe montrant l'évolution de la quantité de nourriture montre expérimentalement qu'il y a convergence ; elle peut de plus être exploitée pour mesurer un taux de convergence. Pour ce faire, une régression linéaire est effectuée sur une partie des dernières données : au temps t on utilise les données enregistrés de $0.5t$ à t. Lorsque le processus a convergé, les paramètres de la droite de régression se stabilisent, et nous proposons de définir le temps de convergence comme l'intersection de la droite et de l'axe des abscisses[4] (voir la figure 3). Numériquement, nous effectuons des régressions et des estimations du temps de convergence jusqu'à ce que ce dernier varie de moins de 0.01%. Le taux de convergence est alors naturellement défini comme l'inverse du temps de convergence.

En utilisant cette méthode, on peut faire des statistiques sur le taux de convergence et sa dépendance en fonction du nombre de fourmis. Pour une taille de population don-

4. Ce calcul est analogue à celui de la constante de temps d'une capacité électrique.

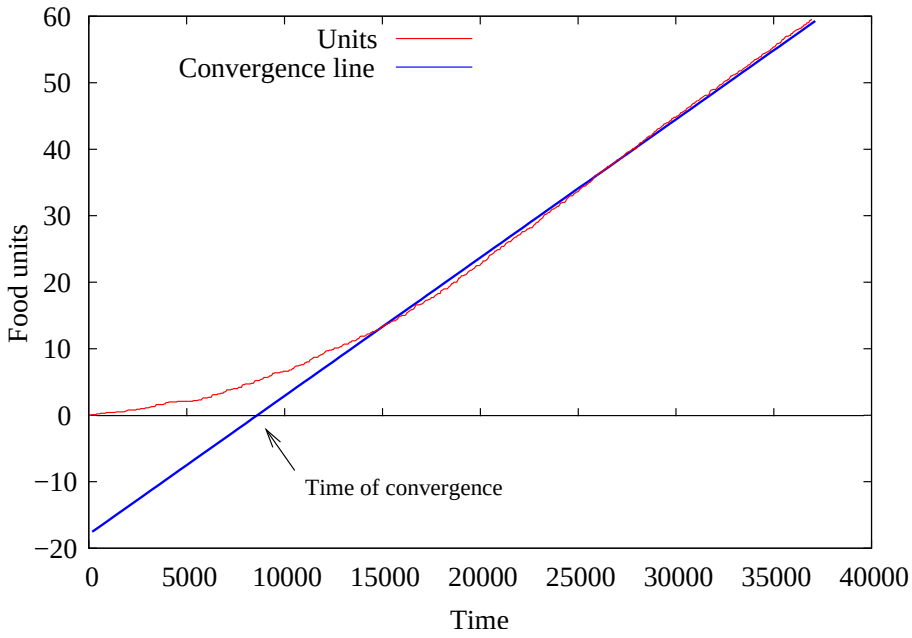

Figure 3. *Mesure du taux de convergence : unités de nourriture correspond à la quantité de nourriture accumulée au nid normalisée par le nombre de fourmis, et la droite de convergence correspond la régression linéaire sur les données.*

née, une expérience consiste en l'exécution de l'algorithme un certain nombre de fois (ici vingt fois) sur un environnement donné avec les mêmes valeurs de paramètres(α, β, ϵ et m). À chaque exécution le taux de convergence est mesuré, et à la fin on calcule le taux moyen ainsi que son écart-type.

A la figure 4, on présente une courbe typique montrant le taux de convergence en fonction de la taille de la population. Pour ces expériences, nous avons fixé $\alpha = 0.7$, $\beta = 0.9999$, $\epsilon = 0.8$, et les $m = 150$ fourmis étaient initialisées au nid. Une telle courbe illustre expérimentalement que quand le nombre de fourmis croît linéairement, le taux de convergences croît de manière super-linéaire. En d'autres termes, si le nombre de fourmis est doublé le taux de convergence croît d'un facteur supérieur à deux. Ce résultat sera analysé dans la partie 4 où nous fournirons des arguments qui expliquent pourquoi le taux de convergence croît de la sorte.

3. Analyse de la convergence

Dans cette partie, nous analysons le modèle de fourmis décrit dans la partie précédente, et nous prouvons sa convergence. Pour comprendre ce que fait le modèle

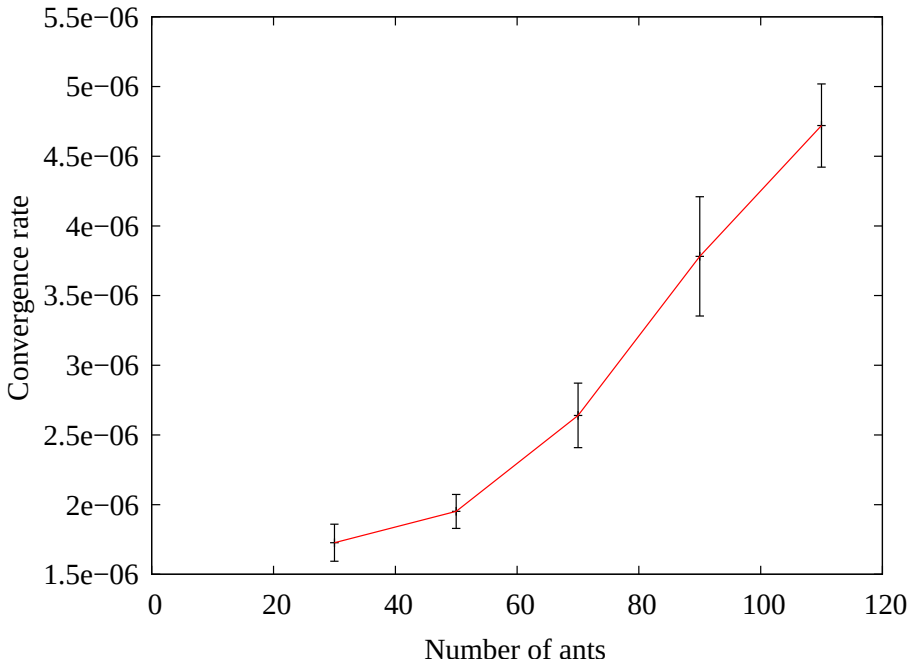

Figure 4. *Taux de convergence en fonction de la taille de la population.*

et pourquoi il y a émergence de chemins, nous allons en premier lieu faire un petit détour par la théorie du contrôle optimal stochastique en temps discret, et plus particulièrement par le cadre des processus décisionnels de markov (PDM). Un PDM est un processus stochastique contrôlé qui vérifie la propriété de Markov avec des récompenses (valeurs numériques) affectées à des états. Plus formellement, un PDM est un quadruplet $\langle S, A, T, R\rangle$ où S est *l'espace des états*, A est *l'espace des actions*, T est la *fonction de transition* et R est la *fonction de récompense*. La fonction T est la distribution de probabilités de transition d'états conditonnellement aux actions : Pour tout états s et s' et action a,

$$T(s, a, s') \stackrel{def}{=} \Pr(s_{t+1} = s' | s_t = s, a_t = a).$$

La fonction $R(s) \in \mathbb{R}$ est la récompense instantanée d'être dans l'état s. Dans le cadre des PDM, le *problème du contrôle optimal* consiste à trouver une *politique*, c'est-à-dire une application $\pi : S \rightarrow A$ des états vers les actions, qui maximise l'espérance de la récompense accumulée à long terme, aussi appelée fonction de valeur de la politique π :

$$V^{\pi}(s) = \mathrm{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t . R(s_t) | s_0 = s\right]. \qquad [3]$$

Nous nous plaçons dans le cadre où l'horizon temporel est infini ; de plus, les récompenses futures sont exponentiellement atténuées par un facteur $\gamma \in (0,1)$ [5]. Pour tout PDM, il a été montré (Puterman, 1994) qu'il existe une unique fonction de valeur optimale V^* qui est le point fixe de l'application suivante sur des fonctions, connu aussi comme opérateur de Bellman :

$$\forall W \in \mathbb{R}^S, [B.W](s) = \max_a \left(R(s,a) + \gamma . \sum_{s'} T(s,a,s').W(s') \right). \quad [4]$$

Une fois la fonction de valeur V^* calculée, une politique optimale π^* peut immédiatement être dérivée comme suit :

$$\pi^*(s) \in \arg\max_a \left(R(s,a) + \gamma . \sum_{s'} T(s,a,s').V(s') \right). \quad [5]$$

L'opérateur de Bellman, B ci-dessus, a un unique point fixe car c'est une *contraction* de facteur de contraction au moins γ : i.e. pour toute paire de fonctions réelles U, U' sur S,

$$\|BU - BU'\| \leq \gamma \|U - U'\|,$$

où $\|.\|$ est la norme infinie sur S : $\|U\| = \max_s |U(s)|$. Une approche standard pour résoudre le problème du contrôle optimal utilise une procédure itérative séquentielle qui s'appelle *Value Iteration* (Puterman, 1994). Elle consiste à initialiser arbitrairement une fonction V^0 et à itérer le processus suivant :

$$\text{Pour tout état } s \in S, \text{ faire : } V^{t+1}(s) \leftarrow [BV^t](s).$$

Grâce à la propriété de contraction, cette séquence est garantie de converger asymptotiquement vers la fonction de valeur optimale V^*, depuis laquelle on peut déduire la politique optimale π^* (cf équation 5). De plus, une telle séquence a un taux de convergence γ, au moins linéaire :

$$\|V^{t+1} - V^*\| \leq \gamma . \|V^t - V^*\| \leq \gamma^{t+1} . \|V^0 - V^*\| \quad [6]$$

Dans (Bertsekas *et al.*, 1989), les auteurs expliquent qu'une version asynchrone de *Value Iteration* :

$$\text{Choisir (aléatoirement) un état } s \in S \text{ et faire : } V(s) \leftarrow [BV](s), \quad [7]$$

convergera aussi vers le point fixe V^*, *tant que tout les états continuent d'être choisis*. Dans le cas asynchrone, on peut réécrire l'équation 6 sous la forme :

$$\|V^{k_t} - V^*\| \leq \gamma \, \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\|, \quad [8]$$

5. Prendre $\gamma < 1$ peut être vu comme une astuce mathématique pour que le critère de performance ci-dessus soit borné.

où $k_0, k_1 \ldots$ est une suite croissante telle que $k_0 = 0$ et que tous les éléments de S (tout les états) sont mis à jours au moins une fois entre les instants k_t et $k_{t+1} - 1$ pour tout t (voir (Bertsekas *et al.*, 1996) p. 27). Chaque fois que tout les états ont été mis à jour, on sait que V s'approche de V^* avec un taux linéaire au moins γ. La preuve de ce résultat repose sur la propriété de contraction. Bien que le but dans (Bertsekas *et al.*, 1989) était de proposer des implantations parallèles efficaces sur des calculateurs parallèles, le nôtre est légèrement différent : nous allons montrer que la résolution du contrôle optimal est compatible avec le paradigme fourmis.

Revenons à notre modèle de fourmis décrit plus haut, nous allons établir un lien entre ce que la population de fourmis calcule et des problèmes de contrôle optimal. En effet, nous allons montrer que les traces de phéromones V_1 et V_2 correspondent chacune à la fonction de valeur d'un problème de contrôle. Ceci constitue notre premier résultat :

Proposition 1 *Considérons le modèle de fourmis décrit dans la partie 2.1. Si le taux d'exploration $\epsilon > 0$, alors les phéromones V_1 (resp. V_2) convergent asymptotiquement vers la fonction de valeur optimal du PDM $M_1 = \langle S, A, T_1, R_1 \rangle$ (resp. $M_2 = \langle S, A, T_2, R_2 \rangle$) avec le facteur d'atténuation β où :*

– *S est l'ensemble des points de la grille auquel on ajoute un « état final ».*

– *A est l'ensemble des quatre mouvements cardinaux (nord, sud, est, ouest).*

– *La transition T_1 (resp. T_2) est définie comme suit : 1) quand on est sur une cellule libre ou sur le nid (resp. sur une cellule libre ou sur la source de nourriture) et qu'on choisit une des quatre directions $a \in A$, la probabilité d'aller dans la direction a est $\alpha + \frac{1-\alpha}{4} = \frac{3\alpha+1}{4}$, tandis que la probabilité d'aller dans une des trois autres directions est $\frac{1-\alpha}{4}$. 2) à partir de tout autre état, i.e. cellules indésirables, source de nourriture et l'« état final » (resp. cellules indésirables, le nid et l'« état final ») il y a, quelle que soit l'action choisie, une probabilité 1 d'arriver à l'« état final », qui est un état absorbant.*

– *La récompense R_1 (resp. R_2) est définie comme suit : pour tout les états s correspondant à une cellule libre ou le nid (resp. une cellule libre ou la source de nourriture), la récompense est 0. Pour l'état correspondant à la source (resp. au nid), la récompense est 1. La récompense est -1 pour les états correspondant aux cellules indésirables et 0 pour l'« état final ».*

La preuve de ce résultat consiste simplement à vérifier que le modèle de fourmis est une version asynchrone de l'algorithme *Value Iteration* pour les problèmes que nous venons de décrire : concrètement il faut vérifier, pour toutes les cellules, que les mises à jours des traces V_1 et V_2 sont identiques à celles que ferait l'opérateur de Bellman

(équation 7). Ceci n'est qu'une simple question de réécriture. Par exemple, pour la mise à jour de $V_i(s)$ quand s est une cellule libre nous avons :

$$
\begin{aligned}
& V_i(s) \leftarrow \beta\left(\alpha \mathrm{max}_1(\mathcal{N}(s)) + (1-\alpha)\mathrm{avg}_1(\mathcal{N}(s))\right) \\
& V_i(s) \leftarrow \beta\left(\alpha \max_{s'\in\mathcal{N}(s)} V_i(s') + \frac{1-\alpha}{4}\sum_{s'\in\mathcal{N}(s)} V_i(s')\right) \qquad [9] \\
\Leftrightarrow \quad & V_i(s) \leftarrow \beta \max_{s'\in\mathcal{N}(s)}\left(\alpha V_i(s') + \frac{1-\alpha}{4}\sum_{s''\in\mathcal{N}(s)} V_i(s'')\right) \\
\Leftrightarrow \quad & V_i(s) \leftarrow \beta \max_{s'\in\mathcal{N}(s)}\left(\left(\alpha + \frac{1-\alpha}{4}\right)V_i(s') \right. \\
& \qquad\qquad \left. + \frac{1-\alpha}{4}\sum_{s''\in\mathcal{N}(s)\setminus\{s'\}} V_i(s'')\right) \\
\Leftrightarrow \quad & V_i(s) \leftarrow \beta \max_{a\in A}\left(\sum_{s'\in S} T_i(s,a,s')V_i(s)\right) \qquad [10]
\end{aligned}
$$

Nous laissons le soin au lecteur intéressé de vérifier l'équivalence dans les autres cas. Enfin la condition $\epsilon > 0$ assure que tous les états sont visités et mis à jour continuellement, ce qui implique la convergence de cette version asynchrone vers la fonction de valeur optimale.

Examinons de plus près ce que cela veut dire et particulièrement pourquoi nous observons une émergence de chemins entre le nid et la source de nourriture lors des simulations. Résoudre M_1 (resp. M_2) signifie trouver la politique qui générera des trajectoires qui évitent (en moyenne) les cellule indésirables pour lesquelles la récompense est -1 et qui atteignent la source de nourriture (resp. le nid) pour lesquelles la récompense est 1 ; à cause du facteur d'atténuation $\beta < 1$, l'optimisation tente aussi de minimiser le temps pour arriver à la source de nourriture (resp. au nid). En d'autres termes, M_1 (resp. M_2) est une formulation naturelle du problème de contrôle suivant : « aller à la source de nourriture (resp. au nid) tout en évitant les cellules indésirables » en supposant qu'il y a du bruit dans les modèles de transition T_1 et T_2. Le niveau de bruit est lié au paramètre α : c'est pour cette raison que nous l'avons appelé paramètre de bruit.

En chaque état, l'action optimale est celle pour laquelle le max est atteint dans l'équation 10 ci-dessus, et il est aisé de voir que cette action optimale est celle qui mènera à la cellule pour laquelle le max est atteint dans l'équation 9 : c'est la cellule avec la plus grande valeur de phéromone. Les valeurs des phéromones convergent asymptotiquement vers la fonction de valeur optimale V_1 et V_2. Par conséquent, les mouvements des fourmis qui sont (cf. 2.1) un mélange de mouvements aléatoires (avec

un poids ϵ) et des mouvements dans la direction la plus concentrée en phéromone (avec un poids $1 - \epsilon$), convergent vers un mélange d'actions aléatoires et d'actions optimales. Plus ϵ est petit, plus net sera le chemin observé quand les phéromones auront convergé (à la limite $\epsilon = 0$ toutes les fourmis suivent la politique optimale). Cependant, plus ϵ est petit, plus le temps que mettront les fourmis à visiter toutes les cellules sera grand, et par conséquent la convergence des phéromones sera ralentie. Ce compromis est typique de la théorie du contrôle optimal et est connu comme le dilemme exploration-exploitation (voir par exemple (Sutton *et al.*, 1998)).

Lors de la description du modèle dans la partie 2.1, nous avons écrit à propos des paramètres α et β qu'ils devaient satisfaire $0 \leq \alpha \leq 1$ et $0 \leq \beta \leq 1$ avec la condition que $\beta < 1$ si $\alpha = 1$. Nous pouvons à présent expliquer cette condition du point de vue du contrôle optimal : choisir le facteur d'atténuation strictement inférieur à 1 dans un PDM garanti que la performance à horizon temporel infini (eq. 3) est bornée et que l'opérateur de Bellman est une contraction. Dès lors qu'il y a du bruit dans le modèle de transition des PDM M_1 et M_2 (i.e. $\alpha < 1$), pour n'importe quelle action, il y a une probabilité 1 d'atteindre en un temps fini l'« état final » absorbant (qui a une récompense 0). Ceci est suffisant pour garantir que le critère de performance est borné et que l'opérateur de Bellman est une contraction. Toutefois, dans le cas purement déterministe ($\alpha = 1$) le facteur d'atténuation β doit être strictement inférieur à 1.

Nous pouvons de plus expliquer l'influence du paramètre de bruit α. Quand α est égal à 1 (et la mise à jour est l'équation 1), il n'y a pas de bruit dans le modèle de transition et les politiques optimales correspondent au chemin le plus court (par rapport à la distance de *Hamming*, voir la figure 2, cas $\alpha = 1$) entre le nid et la source de nourriture. Quand on réduit α, le niveau de bruit augmente et les politiques optimales induisent des trajectoires plus arrondies (elles tentent de rester loin des états indésirables). Quand α continue à décroître, il y a tellement de bruit que la probabilité d'atteindre un état indésirable en essayant d'arriver à la source de nourriture (ou au nid) devient trop grande pour n'importe quelle politique. Par conséquent, le comportement optimal consiste à rester loin des états indésirables sans tenter d'atteindre la source de nourriture (ou le nid) : dans ce cas il est préférable d'avoir une récompense nulle (état libres) qu'une récompense -1 (états indésirables). Ceci explique pourquoi il n'y a pas de chemin sur la figure 2 pour $\alpha = 0.6$. Quand α décroît encore plus et se rapproche de 0, il se produit un phénomène à première vue étrange : des chemins apparaissent de nouveau. Nous pouvons expliquer ceci de deux manières : 1) quand le bruit devient trop grand, l'influence des actions diminue et le contrôleur optimal ne peut même plus rester éloigné des états indésirables ; alors, tel un kamikaze qui sait qu'il va mourir, la stratégie optimale consiste à nouveau à essayer d'atteindre la source de nourriture (ou le nid). 2) À la limite quand $\alpha = 0$ (et que la mise à jour est l'équation 2), les taux de phéromones, qui vérifient une équation du type $V_i(s) = \text{avg}^i(\mathcal{N}(s))$ sont équivalentes à la version discrète d'une fonction harmonique, et il est connu que monter le gradient d'une fonction harmonique peut être utilisé pour la navigation (voir (Connolly *et al.*, 1993) pour plus de détail).

4. Analyse du taux de convergence

Nous avons vu que le modèle de fourmis proposé converge en expliquant qu'il est une instance de calcul asynchrone de points fixes de deux contractions : les traces de phéromones qui résultent des mise à jours locales faites par les fourmis se stabilisent vers les fonctions de valeur optimales de deux problèmes de contrôle, ce qui guide les fourmis entre le nid et la source de nourriture. Le but de cette partie est l'étude du *taux de convergence* de ce processus en fonction de la taille de la population de fourmis. Pour cette analyse, nous allons décrire quelques objets et quelques résultats sur les chaînes de Markov sur les graphes qui expliquent l'observation (faite dans la partie 2.2) que le taux de convergence est super-linéaire : doubler le nombre de fourmis peut accélérer le processus d'un facteur supérieur à deux.

Pour l'analyse du taux de convergence, nous revenons à l'équation 8 qui décrit, en général, le taux de convergence du calcul asynchrone du point fixe d'une contraction. Par souci de clarté nous réécrivons l'équation ici :

$$\|V^{k_t} - V^*\| \leq \gamma \|V^{k_{t-1}} - V^*\| \leq \gamma^t \|V^0 - V^*\|, \qquad [11]$$

Nous rappelons au lecteur que $k_0, k_1 \ldots$ est une suite croissante telle que $k_0 = 0$ et que tout les éléments de s (tout les états) sont mis à jour au moins une fois entre k_t et $k_{t+1} - 1$ pour tout t. On peut constater donc que le taux de convergence est clairement lié à la variable aléatoire k_t : plus k_t croît lentement, plus le processus converge vite. Comme nous nous intéressons au taux de convergence en fonction de la taille m de la population, nous allons expliciter cette dépendance en écrivant k_t^m. Il nous faut donc étudier $k_{t+1}^m - k_t^m$.

La bonne nouvelle au sujet de $k_{t+1}^m - k_t^m$ est que c'est un objet connu de la littérature probabiliste. Considérons l'espérance $E[k_{t+1}^1 - k_t^1]$ pour le cas $m = 1$: c'est le temps moyen nécessaire à une fourmi pour visiter toutes les cellules. Plus formellement, si nous considérons l'environnement comme un graphe G (il y a un nœud pour chaque cellule et une connexion entre deux nœuds voisins), c'est le temps moyen nécessaire à une chaîne de Markov (qui décrit les positions de cette fourmi) pour visiter tout les nœuds du graphe G, et cette quantité est appelée le *temps de couverture du graphe G par la chaîne de Markov* (Aldous *et al.*, 1996). Dans le cas général $m > 1$, $E[k_{t+1}^m - k_t^m]$ est le temps moyen pour que plusieurs marches aléatoires parallèles visitent tous les nœuds du graphe G, et cette quantité s'appelle le *temps de couverture du graphe G par m chaînes de Markov parallèles* (Aldous *et al.*, 1996).

La mauvaise nouvelle au sujet de $k_{t+1}^m - k_t^m$ est que dans le cas *général* il est très difficile de calculer le temps de couverture d'un graphe par une chaîne de Markov donnée ayant une distribution initiale donnée : il est calculable en temps exponentiel et la possibilité que le calcul puisse être approché en temps polynomial est une question ouverte (Feige *et al.*, 2003). Il est plus difficile de calculer le temps de couverture d'un graphe pour plusieurs chaînes de Markov. Pour notre modèle de fourmis, le calcule est encore plus difficile puisque les chaînes de Markov et la distribution des fourmis varient en fonction du temps de façon non triviale : les probabilités de tran-

sitions dépendent des traces de phéromones qui elles mêmes sont mises à jour par les chaînes de Markov. Il y a donc peu d'espoir qu'on puisse estimer *précisément* le taux de convergence de notre modèle de fourmis. Une analyse asymptotique générale de la convergence (dans laquelle on considère que a) les phéromones ont convergé et b) les fourmis ont atteint une distribution stationnaire) pourrait être poursuivie et constitue un sujet pour des recherches futures.

Néanmoins, en consultant la littérature sur le temps de couverture, nous avons trouvé le résultat intéressant suivant (Aldous *et al.*, 1996) :

Proposition 2 *Sur un graphe régulier*[6] *à n nœuds, considérons m marches aléatoires indépendantes et équilibrées*[7]*, chacune commençant à des nœuds distribués aléatoirement uniformément sur le graphe. Appelons C^m le temps auquel tout les nœuds du graphe ont été atteints par une marche aléatoire*[8]*. Alors à mesure que la taille du graphe $n \to \infty$ et tant que le nombre de marches vérifie $m \geq 6 \log n$, nous avons :*

$$E\left[C^{[m]}\right] \leq \frac{(25 + o(1))n^2 \log^2 n}{m^2}.$$

Le point intéressant est la dépendance en $\frac{1}{m^2}$: ceci implique que (à mesure que $n \to \infty$ et $m \geq 6 \log n$) le temps de couverture est borné par une fonction qui est inversement quadratique en le nombre de marches. Sur des graphes simples comme un cycle de taille n, (Aldous *et al.*, 1996) explique que la borne ci-dessus est fine : dans ces cas, le temps de couverture est inversement quadratique en le nombre de marches.

En utilisant la discussion ci-dessus, nous pouvons "traduire" cette proposition en quelque chose ayant un sens dans le cadre de notre modèle de fourmis :

Proposition 3 *Considérons le modèle de fourmis décrit dans la partie 2.1 sur un environnement torique avec n cellules, avec un taux d'exploration $\epsilon = 1$ et une initialisation des fourmis uniformément distribuée sur la grille. À mesure que la taille de l'environnement $n \to \infty$ et tant que le nombre de fourmis vérifie $m \geq 6 \log n$, le temps que mettent les traces de phéromones pour réduire d'un facteur β leur distance à la limite est borné par $\frac{(25+o(1))n^2 \log^2 n}{m^2}$ qui est une fonction inversement quadratique en le nombre m de fourmis.*

L'hypothèse de l'environnement torique permet d'avoir un graphe régulier, et $\epsilon = 1$ permet d'avoir des marches aléatoires équilibrées. Notons que l'initialisation uniforme des fourmis correspond à la distribution stationnaire des marches, de sorte que la distribution des fourmis est stationnaire au cours du temps. Notre proposition n'est donc

6. Un graphe régulier est un graphe dont les nœuds ont tous le même degré, i.e chaque nœud est connecté au même nombre de nœuds.
7. Une marche aléatoire équilibrée sur un graphe est une marche aléatoire dont les transitions sont des distributions uniformes sur les voisins.
8. $E[C^m]$ est donc le temps de couverture du graphe par ces marches aléatoires parallèles.

qu'un corollaire de la proposition 2. La borne est fine quand l'environnement est un graphe cycle de taille n, c'est à dire un long couloir cyclique : dans ce cas, le taux de convergence peut être quadratique en le nombre de fourmis. Nous avons dans ce cas un taux de convergence super-linéaire. Les expériences que nous avons effectuées suggèrent que la super-linéarité se produit pour d'autres valeurs de ϵ et dans d'autres environnements. Étendre le résultat ci-dessous à des situations plus générales, est l'objet de recherches futures.

5. Extension du modèle au cas continu

Jusqu'ici nous avons considéré un environnement discret. Le but de cette partie est de montrer que notre modèle peut être adapté à une tâche de fourragement dans un espace continu, dans laquelle les fourmis peuvent se déplacer dans toutes les directions $\theta \in (0, 2\pi)$. Pour ce faire, notre approche sera particulièrement naturelle : nous allons décrire une variante continue du problème de navigation dans le cadre du contrôle optimal à temps continu, expliquer que ce problème continu peut être approché par un algorithme distribué et asynchrone, et en déduire le modèle fourmi correspondant. Bien que nous l'ayons présenté dans l'ordre inverse, c'est d'ailleurs par une démarche analogue que nous avons construit le modèle discret présenté précédemment. Ainsi, cette partie a aussi pour but d'illustrer la méthodologie que nous suivons pour construire des algorithmes fourmis. Pour ne pas compliquer inutilement notre travail, nous nous restreignons ici à la moitié du problème de fourragement : trouver des chemins vers une source de nourriture. L'autre moitié peut être implémentée de la même manière que ce que nous avons fait pour le cas discret : utiliser un état interne pour chaque fourmi et basculer entre les deux problèmes de contrôle sous-jacents : « aller vers la source de nourriture »/« rentrer au nid ».

Un modèle naturel pour la navigation continue est maintenant décrit. Considérons un système défini au temps t par sa position dans les zones libres d'un environnement, i.e. son état $x\,(t) \in \bar{\Omega}$ (l'espace d'états) où $\bar{\Omega} \subset \mathbb{R}^2$ est la fermeture d'un ouvert Ω et où $\partial\Omega$ est son bord ($\bar{\Omega} = \Omega \cup \partial\Omega$). Les bords de l'environnement sont décomposés en deux ensembles : $\partial\Omega = \mathcal{B} \cup \mathcal{F}$: $\mathcal{B}$ est l'ensemble des "bords indésirables" et $\mathcal{F}$ est l'ensemble des "bords nourriture". A chaque instant, le système est contrôlé par une commande de direction $\theta(t) \in (0, 2\pi)$ (l'espace de contrôle). La dynamique de ce système est régie par l'équation différentielle stochastique suivante :

$$dx = \overrightarrow{u}(\theta(t))\; dt + \sigma\; dw$$

où w est un mouvement brownien de dimension 2, σ est une constante positive qui correspond au niveau de bruit dans l'environnement, et $\overrightarrow{u}(\theta)$ est un vecteur unitaire de direction $(0, 2\pi)$ (c'est-à-dire que le contrôle est de vitesse constante unitaire). Nous considérons le cas où l'horizon temporel est infini. Pour tout état initial $x(0)$, pour toute loi de contrôle $\theta\,(.)$ et pour la trajectoire (aléatoire) $x\,(.)$ qui en résulte, nous notons T le temps pour que la trajectoire $x\,(.)$ sorte de Ω, avec la convention que $T = \infty$ si la trajectoire reste indéfiniment dans Ω. Nous utilisons une récompense pour

évaluer la qualité des trajectoires : en général, il est usuel de définir une récompense tout le long de la trajectoire $r : \Omega \rightarrow \mathbb{R}$ et une récompense terminale $R : \partial\Omega \rightarrow \mathbb{R}$ pour le moment où la trajectoire atteint le bord. Pour notre problème de navigation, nous posons $r(x) = 0$ ($\forall x \in \Omega$), $R(x) = -1$ sur les "bords indésirables" ($\forall x \in \mathcal{B}$) et $R(x) = 1$ sur les "bords nourriture" ($\forall x \in \mathcal{F}$). De manière similaire au cas discret, nous introduisons une fonction de valeur optimale qui correspond à la quantité maximale de renforcement sur les trajectoires avec un facteur d'actualisation ($\gamma \in (0, 1)$) :

$$\begin{aligned} J^*(x) &= \max_{\theta(.)} E_{\theta(.)} \left[\int_0^T \gamma^t r(x(t))\, dt + \gamma^T R(x(T)) \right] \\ &= \max_{\theta(.)} E_{\theta(.)} \left[\gamma^T R(x(T)) \right] \end{aligned}$$

où $E_{\theta(.)}$ est l'espérance sur les trajectoires induites par la loi de contrôle $\theta(.)$. Le terme intégral disparaît entre la première et la deuxième ligne car $r(\cdot) = 0$. Vu que la fonction $t \mapsto \gamma^t$ est décroissante et étant donnée notre définition de la récompense terminale (-1 sur les "bords indésirables" et 1 sur les "bords nourriture"), maximiser cette quantité revient à la fois à 1) maximiser le temps pour toucher un "bord indésirables" et 2) minimiser le temps pour atteindre un "bord nourriture".

La théorie du contrôle optimal montre que la fonction de valeur ci-dessus vérifie une équation aux dérivées partielles, l'équation de Hamilton-Jacobi-Bellman :

$$J^*(x) \ln(\gamma) + \max_{\theta \in (0,2\pi)} \{\nabla J^*(x) \cdot \overrightarrow{u}(\theta)\} + \frac{\sigma^2}{2} \Delta J^*(x) = 0 \qquad [12]$$

pour $x \in \Omega$ avec les conditions au bord $\forall x \in \partial\Omega, J^*(x) = C(x)$. Nous avons noté ∇J^* le gradient de J^* et ΔJ^* le Laplacian de J^*. De manière analogue au cas discret, le contrôle optimal (la direction optimale $\theta^*(x)$ lorsqu'on est à la position x) est celle pour laquelle le max est atteint. Dans notre cas, il est facile de voir que le contrôle optimal $\theta^*(x)$ est la direction qui monte le gradient $\nabla J^*(x)$ de la fonction de valeur le plus rapidement : $\overrightarrow{u}(\theta^*(x)) = \frac{\nabla J^*(x)}{\|\nabla J^*(x)\|}$. En effet, $\nabla J^*(x) \cdot \overrightarrow{u}(\theta)$ est maximal lorsque θ est dans la direction de plus haute pente de $\nabla J^*(x)$.

En pratique, il n'est pas possible de calculer des solutions analytiques à l'équation 12, et une approche standard consiste à construire un schéma aux différences finies. Pour ce faire, nous suivons la démarche de (Kushner *et al.*, 1992). Etant donnée une résolution $\delta > 0$, nous construisons une grille Σ^δ et son bord $\partial\Sigma^\delta$ sur le domaine Ω. Pour cette résolution, la fonction de valeur J est approchée par une fonction J^δ définie sur $\Sigma^\delta \cup \partial\Sigma^\delta$ et le contrôleur optimal est celui qui remonte le gradient d'une interpolation (par exemple une triangulation) de J^δ sur Ω.

Un tel processus de discrétisation a d'intéressantes propriétés : on peut montrer que l'approximation J^δ de la fonction de valeur J^* est la fonction de valeur d'un problème de contrôle à temps discret, c'est-à-dire d'un certain PDM. Ainsi nous retombons sur

le cas PDM que nous avons exploité dans les sections précédentes. En conséquence, il est facile de construire un modèle fourmi qui calcule J^δ : il suffit que ce modèle implémente une version asynchrone de l'algorithme *Value Iteration*. La mise à jour des phéromones doit correspondre à l'opérateur de Bellman, opérateur que nous décrivons maintenant.

Notons $\cos^+$, $\cos^-$, $\sin^+$ et $\sin^-$ les parties positives et négatives des fonctions $\cos$ et $\sin$: $\cos^\pm(\theta) = \max(\pm\cos\theta, 0)$ and $\sin^\pm(\theta) = \max(\pm\sin\theta, 0)$. De plus, définissons les probabilités de transition d'un point de la grille de discrétisation (x, y) vers ses quatre voisins lorsqu'on choisit d'aller dans la direction θ :

$$\left\{ \begin{array}{l} p_\theta\left[(x,y),(x\pm\delta,y)\right] = \frac{1}{N_\theta}\left[\frac{\sigma^2}{2} + \delta\cos^\pm\theta\right] \\ p_\theta\left[(x,y),(x,y\pm\delta)\right] = \frac{1}{N_\theta}\left[\frac{\sigma^2}{2} + \delta\sin^\pm\theta\right] \end{array} \right. \quad [13]$$

où $N_\theta = \delta(\cos^+\theta + \cos^-\theta + \sin^+\theta + \sin^-\theta) + 4\sigma^2/2 = \delta(|\cos\theta| + |\sin\theta|) + 2\sigma^2$ peut être vu comme un facteur qui assure que la somme des probabilités de transition est 1. Enfin, notons $\tau(\theta) = \frac{\delta^2}{N_\theta}$, quantité qui peut être interprétée comme le temps requis pour aller du point (x, y) vers l'un de ses voisins lorsqu'on va dans la direction θ. Alors, on peut montrer que l'opérateur de Bellman associé à J^δ est :

$$B^\delta[W](x,y) = \gamma^{\tau(\theta^W_{x,y})} \sum_{(x',y')\in\Sigma^\delta} p_{\theta^W_{x,y}}\left[(x,y),(x',y')\right] W(x',y')$$

pour $(x, y) \in \Sigma^\delta$ et $B^\delta[W] = C$ sur $\partial\Sigma^\delta$, où $\theta^{J^\delta}_{x,y}$ est l'angle qui correspond à la direction de plus grande pente de J^δ au point (x, y) lorsqu'on considère une interpolation linéaire par morceaux de J^δ sur Ω. La notation $\theta^{J^\delta}_{x,y}$ peut paraître un peu lourdre, mais nous l'utilisons car il est important de se souvenir que l'angle correspondant au contrôle optimal dépend des coordonnées (x, y) et de la fonction de valeur J^δ.

Maintenant que l'opérateur de Bellman B^δ est explicité, il est facile de construire un algorithme fourmi qui implémente le calcul de son point fixe. C'est ce que nous faisons maintenant. Les fourmis sont situées dans l'environnement original (elles ont des coordonnées qui peuvent prendre des valeurs continues) et s'y déplacent en mettant à jour des taux de phéromones qui correspondent aux valeurs de la fonction J^δ. Plus précisément, à chaque instant, chaque fourmi fait exactement deux choses :

– Elle met à jour la valeur de phéromone $J^\delta(x, y)$ du point (x, y) qui lui est le plus proche sur la grille comme suit :

$$J^\delta(x,y) \leftarrow [B^\delta J^\delta](x,y)$$

c'est-à-dire en utilisant uniquement les valeurs des phéromones des voisins du point (x, y).

– Elle fait un déplacement de longueur $l \in \mathbb{R}$:

- Avec probabilité ϵ (le taux d'exploration), elle se déplace dans une direction choisie aléatoirement uniformément dans $(0, 2\pi)$.

- Avec probabilité $1 - \epsilon$, elle se déplace dans la direction qui monte le plus vite le gradient de l'interpolation locale des phéromones J^δ (autrement dit elle suit le contrôle optimal correspondant à J^δ).

Comme nous avons simplement modélisé une moitié du problème (la recherche de la source de nourriture), nous supposons de plus que les fourmis qui arrivent à une source de nourriture sont réinitialisées au nid.

Une simulation typique de ce modèle est illustrée à la figure 5, dans un environnement avec un nid et deux sources de nourriture. Toutes les fourmis sont initialisées au nid. Comme attendu, on observe l'émergence d'un chemin entre le nid et l'une des source de nourriture. Ce chemin est "renforcé" au fur et à mesure que le temps passe.

Dans ce cas, il est également possible de faire une analyse de la convergence. Si le taux d'exploration ϵ est strictement positif, les fourmis passent indéfiniment à proximité de chacun des points de la grille de discrétisation. La population constitue donc une version asynchrone de l'algorithme *Value Iteration* et les phéromones convergent vers la solution approchée J^δ. On peut également caractériser le taux de convergence en utilisant une notion généralisée du temps de couverture. Dans le cas que nous venons de décrire, où les fourmis se déplacent dans l'espace continu et mettent à jour la valeur de phéromone du point de la grille le plus proche, le processus dynamique qui décrit la manière avec laquelle les points de la grille sont mis à jour n'est plus une simple chaîne de Markov mais une chaîne de Markov cachée : les points de la grille qui sont mis à jour sont une fonction de la position (réelle et continue) de chaque fourmi, qui est elle-même une chaîne de Markov dans l'environnement continu. Le temps de couverture qu'il faut considérer est donc celui d'une chaîne de Markov cachée. Malheureusement, il n'y a à notre connaissance pas de résultat dans la littérature concernant ce type de temps de couverture. Si nous avons pu observer expérimentalement que nous avons ici encore un taux de convergence superlinéaire, nous n'avons pas trouvé dans la littérature de résultat théorique correspondant.

6. Discussion et conclusion

Nous avons présenté une classe de modèles de fourmis qui peut être liée au cadre du contrôle optimal, et nous avons prouvé leur convergence en un certain sens. Nous avons aussi analysé le taux de convergence de ces modèles en fonction de la taille de la population : nous avons montré, à l'aide de simulations et analytiquement (pour le cas discret) que le taux de convergence est super-linéaire en le nombre de fourmis. À première vue, la super-linéairité peut apparaître comme une propriété très intéressante. Il faut toutefois la relativiser : la super-linéairité est due au fait que de petites populations sont *particulièrement* inefficaces pour explorer l'espace et faire converger les traces de phéromones. En fait, l'analyse de convergence que nous avons faite suggère clairement (voir les équations 6 et 8) que la méthode la plus rapide pour calculer la fonction de valeur optimale est la méthode synchrone : à chaque pas de temps, la va-

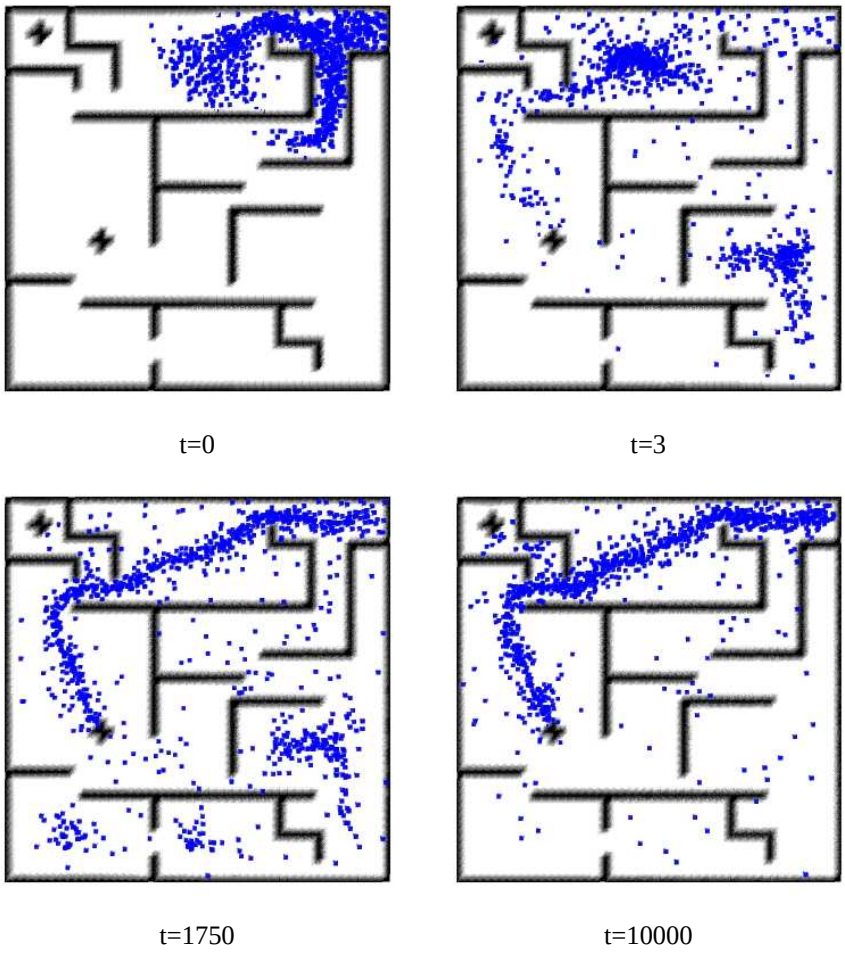

Figure 5. *Expérience typique montrant la convergence et l'émergence d'un chemin : six "photos" d'une exécution de l'algorithme continu dans un environnement contenant* 50×50 *points grilles et* 500 *fourmis*

leur s'approche de sa limite à un taux linéaire γ. La lenteur relative du modèle fourmi doit être considérée comme le prix de la décentralisation.

Le modèle que nous avons présenté dans cet article est très flexible. Il peut incorporer plusieurs variations. On peut, par exemple, considérer que l'environnement contient plusieurs sources de nourritures chacune en quantité finie, qui décroît en fonction des visites des fourmis. Ces quantités peuvent être incorporées dans le modèle du contrôle optimal à travers la fonction récompense aux états sources. La récompense

évolue alors au cours du temps, et les traces de phéromones, qui sont constamment mises à jours, suivront cette fonction de valeur optimale « changeante ». On peut aussi utiliser différents paramètres pour le voyage aller et le voyage retour des fourmis : ceci pourrait modéliser des stratégies différentes selon que les fourmis transportent ou non de la nourriture. Enfin, si l'environnement est statique, on peut faire décroître lentement le taux d'exploration ϵ vers 0, de sorte que les fourmis convergent finalement vers la politique optimale déterministe. Étudier de bons moyens de « figer » le comportement des fourmis via le paramètre d'exploration ϵ fait l'objet de recherches futures.

Il est facile de voir qu'on peut construire des algorithmes fourmis du même genre que celui que nous avons présenté pour résoudre n'importe quel problème de contrôle optimal. Tant que le problème est formulé comme un PDM, nous savons qu'il peut être résolu par une version asynchrone de *Value Iteration*, et construire le modèle de fourmis correspondant est immédiat : il suffit que ces fourmis évoluent dans l'environnement et mettent à jour une phéromone qui joue le rôle de la fonction de valeur partout où elles passent. Si l'on veut satisfaire la contrainte que « les fourmis doivent prendre leurs décisions en utilisant uniquement l'information locale », alors cette approche marchera tant que les transitions, dans le PDM, sont *locales* dans l'espace d'états. Nous pouvons aller encore plus loin : toute notre analyse (la convergence et le taux de convergence) repose sur le propriété de « contraction » qui est vérifiée par l'opérateur de Bellman. Ceci suggère que tout problème qui implique le calcul du point fixe d'une contraction (par exemple, on trouve des contractions dans des problèmes tels que la recherche du zéro d'une fonction, ou l'optimisation (Luenberger, 1989)) sur un espace fini a une solution naturelle par un algorithme fourmi : les fourmis évoluent dans l'espace fini et effectuent de manière distribuée et asynchrone le calcul du point fixe de cette contraction.

Pour finir, nous faisons référence à des travaux récents qui sont très proches de ce que nous avons fait ici.

– Dans (Simonin, 2005), l'auteur présente un algorithme multi-agent qui calcule le plus court chemin entre le nid et n'importe quel point d'une grille sous la forme d'une implantation asynchrone de l'algorithme de Bellman-Ford : la mise à jour locale est de la forme $U(x) \leftarrow 1 + \min_{x' \in \mathcal{N}(x)} U(x')$. Modulo le changement de variable $U \leftrightarrow \frac{\log(V)}{\log(\beta)}$, ceci est équivalent à notre mise à jour des traces de phéromones avec $\alpha = 1$ (eq. 1). L'auteur utilise ce mécanisme comme une base pour construire un système de fourragement distribué efficace (il utilise un mécanisme original pour l'exploration et la coopération via une seconde trace).

– Dans (Panait *et al.*, 2004), les auteurs considèrent la problématique du fourragement et proposent d'identifier, comme nous le faisons ici, la fonction de valeur de problèmes de contrôle à des taux de phéromones. Ils utilisent des versions asynchrones de *Value Iteration* ainsi que certaines variantes algorithmiques (dites d'apprentissage par renforcement) pour résoudre le problème du contrôle optimal (Sutton, 1988). Ils présentent de nombreuses expériences qui montrent que ce genre de systèmes est robuste dans des environnements contenant des obstacles, où la position du nid et des

sources de nourriture varient au cours du temps, et s'il y a plusieurs chemins à trouver (ils utilisent, comme nous, un taux de phéromone par chemin à trouver).

Il y a des différences importantes entre ces travaux et le nôtre :

– Ces travaux se concentrent sur des problèmes de navigation déterministe alors que nous venons d'argumenter qu'en principe, on peut résoudre n'importe quel problème de contrôle optimal stochastique (voire même tout calcul du point fixe d'une contraction) par un algorithme de type fourmi.

– A notre connaissance, il n'y a pas d'arguments concernant la convergence pour les algorithmes de (Panait *et al.*, 2004) tandis qu'une version étendue de (Simonin, 2005) contient une preuve qui repose sur le même genre d'arguments que celui que nous donnons dans ce papier (Simonin, 2006) .

– Enfin, la super-linéarité du taux de convergence est observée expérimentalement dans (Simonin, 2006) mais pas dans (Panait *et al.*, 2004), et dans aucun cas elle n'est étudiée théoriquement.

En d'autre termes, notre travail, peut être considéré comme une généralisation et un approfondissement théorique de (Panait *et al.*, 2004; Simonin, 2005).

7. Bibliographie

Aldous D., Fill J., *Reversible Markov Chains and Random Walks on Graphs*, Monograph in preparation, 1996.

Beni G., Wang. J., « Swarm intelligence », *in* R. press. (ed.), *Seventh Annual Meeting of the Robotics Society of Japan*, p. 425-428, 1989.

Bertsekas D., Tsitsiklis J. N., *Parallel and Distributed Computation : Numerical Methods*, Prentice hall, 1989.

Bertsekas D., Tsitsiklis J. N., *Neuro-Dynamic Programming*, Athena Scientific, 1996.

Bonabeau E., Dorigo M., Theraulaz G., *Swarm intelligence : from natural to artificial systems*, Oxford University Press, Inc., New York, NY, USA, 1999.

Connolly C., Grupen R., « On the Applications of Harmonic Functions to Robotics », *Journal of Robotic and Systems*, vol. 10, n° 7, p. 931-946, October, 1993.

Dorigo M., Blum C., « Ant colony optimization theory : a survey », *Theoretical Computer Science*, vol. 344, n° 2-3, p. 243-278, 2005.

Feige U., Rabinovich Y., « Deterministic approximation of the cover time », *Random Struct. Algorithms*, vol. 23, n° 1, p. 1-22, 2003.

Kohonen T., *Self-Organization and Associative Memory*, Springer-Verlag, 1988.

Kushner H., Dupuis P., *Numerical Methods for Stochastic Control Problems in Continuous Time*, Springer Verlag, 1992.

Luenberger D., *Linear and nonlinear programming*, Addison-Wesley, New York, 1989.

Panait L., Luke S., « A Pheromone-Based Utility Model for Collaborative Foraging », *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

Puterman M., *Markov Decision Processes*, Wiley, New York, 1994.

Simonin O., « Construction of numerical potential fields with reactive agents », *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

Simonin O., Construction of numerical potential fields with reactive agents, an efficient solution for collaborative foraging, Technical report, INRIA, 2006.

Sutton R., « Artificial intelligence as a control problem : Comments on the relationship between machine learning and intelligent control », *IEEE International Symposium on Intelligent Control*, p. 500-507, 1988.

Sutton R., Barto A., *Reinforcement Learning, An introduction*, BradFord Book. The MIT Press, 1998.